修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院　　情報理工学研究科　情報・ネットワーク工学専攻　博士前期課程		
氏　　名	ZHANG SHUYU	学籍番号	1931086
論 文 題 目	AUC Maximization in Deep Neural Network Learning for Imbalanced Classification Problems （AUC 損失を用いた深層学習による不均衡データ分類器）		

要　　旨

機械学習やデータサイエンスにおける不均衡データとは，クラス数の分布に大きな偏りがあるデータであり，異常検出などの問題で見られる．異常検出において正常品が 999 個，異常品が 1 個のとき，全てのサンプルを正常品に分類すると分類精度が 99.9%である．この時分類精度は適当なパフォーマンス評価を与えない．その代わりに正例と負例に対して，正例に対する識別器の出力が負例より高い確率を表す AUC が評価基準として使われる．

本研究の目的は不均衡なデータに対して AUC の最大化を目指す分類モデルを提案することである．機械学習モデルは損失関数の最小化を目標として最適化を行う，AUC を最大化するためマイナス AUC を損失関数として，深層学習モデルを構築する．AUC は微分不可のため，シグモイド関数を用いた微分可能の近似を用いて，勾配降下法に基づく最適化法が適用できる AUC 損失を提案した．そして AUC 損失の計算量を減らすためミニバッチ学習とサンプリング法を用いて，局所解を避けると同時に計算時間の削減を実現する．また，２クラス分類問題の評価基準である AUC を重み付け平均を用いて多クラス AUC に拡張した．この多クラス AUC を損失関数として最先端の深層学習モデルを用いて多クラス画像分類を行う．

色々な不均衡率を持つリアルなデータセットと合成データで AUC 損失を用いた深層学習モデルの訓練を行った．ニクラス分類について，提案手法は，従来のロジスティクス損失とオーバーサンプリングを用いたモデルと比べて，不均衡なデータセットではより高いパフォーマンスを得た，均衡なデータセットでも同等のパフォーマンスを得た．その上，提案手法は，ロジスティクス損失とほぼ同じレベルの計算時間を実現できた．提案手法は，不均衡なデータによく使われたオーバーサンプリング法より計算時間がかなり短い．多クラスな画像分類について，提案法は，不均衡なデータセットでは多クラスロジスティクス損失より高い AUC を示した．同じく不均衡なデータセットのための Focal Loss と比べて匹敵なパフォーマンスを示した．その上，提案手法はハイパーパラメータの数が Focal Loss より少ないため使いやすい．

AUC Maximization in Deep Neural Network Learning for Imbalanced Classification Problems

鷲沢研究室

1931086

張 書宇

指導教員：鷲沢 嘉一 准教授

2020 年 01 月 24 日

Contents

Chapter 1

Introduction

Deep learning has been rapidly applied to many applications such as computer vision, speech recognition, natural language processing, medical diagnosis, bioinformatics, and text classification [1, 2]. Nowadays, we unconsciously receive benefit of machine learning or artificial intelligence (AI) technology in daily life.

In abnormality detection problems, such as earthquake detection, epilepsy detection, data are often imbalanced. These problems are considered as binary classification problems, which contain positive and negative data points. The prior probability of each class is often imbalanced, e.g., earthquake only lasts a short time whereas most observed data is irrelevant to the prediction of earthquake. In binary situation, if there are only 1% of positive samples, a classification rule which simply classifies all sample into negative class will achieve accuracy of 99%, however, this cannot be accepted. Recent research makes it clear that imbalance datasets raises two problems: 1) training is inefficient, as most data are easy negatives that contribute no useful learning signal; and 2) easy negatives can overwhelm training and lead to degenerate models [3].

To solve the problem, several methods have been proposed for imbalanced classification problems. Resampling methods such as over-sampling (duplicating the negative samples), under-sampling (deleting the positive samples) and the synthetic minority oversampling technique (SMOTE) [4] are often used to make the dataset to be balanced. But those methods are effected by outliers and easy to get overfitting. The online hard example mining (OHEM) [5] screens out a few samples

with greater loss and then put them back into the model for retraining. The cost-sensitive learning imposes an additional cost on misclassification of the minority class [6]. These penalties can bias the model to pay more attention to the minority class. The cost-sensitive loss reweights the training loss such as the focal loss [3] and the cross entropy focal loss (CEFL) [7].

In this paper, we are going to propose another loss function to address imbalanced classification problem. In order to address class imbalanced problems, the area under curve (AUC) of the receiver operating characteristic (ROC) curve has been established [8, 9]. ROC curve is a plot of the true positive rate and false positive rate. We can tune a threshold value for binary problems, hence we can control the trade-off between the false positive rate and true positive rate. By varying the threshold, the classifier behaves from lower sensitivity (higher specificity, lower false positive) to higher sensitivity (lower specificity, higher true positive). AUC takes a value between 0 and 1, the higher AUC value means the better model, and 0.5 is the chance level. If the classifier perfectly classifies data accurately, ROC becomes the unit function, and its AUC value is 1. AUC is often used to evaluate imbalanced problems, abnormality detection or outlier detection problems (positive-unlabeled data).

The advantages of using AUC are 1) suitable for imbalanced problems; 2) robust for outliers and mislabeling data; and 3) finding the optimal decision boundary for all possible thresholding values. As we mentioned, AUC takes into account from the lowest to the highest sensitivity cases. Thus in the case of imbalanced data, AUC measures the performance of the classifier properly. The convex loss functions such as the squared loss and hinge loss are sensitive to outliers and mislabeling data due to its convexity. However, AUC criterion is not convex, and is robust to outliers. The cross-entropy or logistic loss is an approximation of the empirical misclassification, and misclassified training samples have almost the same cost, especially for samples far from the decision boundary. On the other hand, AUC loss avoids misclassification for all possible thresholding values.

Several recent researches have attempted to design classifiers that maximize AUC directly. Since AUC is non-continuous and non-differentiable, the main problem is optimizing model evaluating by AUC. Recent work [10] shows that AUC is derivable under an assumption that distribution of each class is the standard normal distribution. Also an online optimizing method [11] approximates AUC

by the hinge loss, and derives online AUC maximization algorithm. [12] approximates AUC by a quadratic function, and derives a one-path optimization algorithm. There are also researches using a parametric or stochastic approach to maximize AUC [13–15].

We focus on neural networks for imbalanced classification problems using an AUC maximization criterion. Deep learning methods and neural networks are originally developed for classification tasks, and trained to minimize the empirical error expressed as logistic loss, hinge loss or squared loss. With these conventional loss functions, their performances are not always appropriate, especially when the datasets are heavily imbalanced. To the best of author's knowledge, AUC criterion has not been used for the loss function of deep learning. In the neural network model, the loss function we proposed here is simply minus AUC, optimizing the loss function can directly maximize AUC. We formulate AUC by the Stieltjes integral and approximate it by using the sigmoid with a scale parameter. Since the approximated AUC loss is differentiable, it is optimized by the gradient descent method and applied to the training process of the neural network. We also extend the two-class AUC to multiclass form to solve the imbalanced multiclass image classification problem by calculating the weighted average of one-versus-all AUC [16] of each class. Although the problem due to non-linearity of the approximated AUC is remaining, the deep neural network model itself is non-convex, and has many local minima. Recent study suggests that the quality of local minima tends to improve toward the global minimum value as depth and width of network increase in the case of the squared loss [17].

The rest of the thesis is organized as follows. Chapter 2 introduces some related works on addressing imbalanced data, some appropriate performance measurements in imbalanced situation. We also introduce the most popular deep learning architecture for structure data and image classification we used in the experiment. Chapter 3 introduces the proposed AUC loss and the sampling method to reduce run time. Chapter 4 presents our experimental results comparing with the other methods mentioned in Chapter 1. Chapter 5 concludes our work.

Chapter 2

Related Work

2.1 Imbalanced Classification

In machine learning and data science area, some data has imbalanced class distribution where the number of observations belonging to one class is significantly lower than those belonging to the others. The model developed using conventional machine learning algorithms trained with those data could be biased and inaccurate. This happens because most algorithms are usually designed to improve classification accuracy by reducing the loss function. Thus, the class distribution is not considered, since classification accuracy is no longer a reliable performance measurement in imbalanced situation. The conventional model evaluation methods also do not accurately measure model performance when the training samples are imbalanced. Some algorithms have a bias towards classes which have more samples. They tend to only learn the features of the majority class and the features of the minority class are treated as noise or totally ignored. Samples from minority class have a high probability of misclassification.

This section describes various approaches for solving such class imbalance problems. There are mainly two strategies to deal with imbalanced data: 1) resampling; and 2) improving the classification algorithms The resampling method is balancing classes in the training data before providing the data as input to the algorithm. Improving algorithms have many approaches, such as ensembling models and cost-sentive learning.

2.1.1 Resampling

The main objective of balancing classes is to either increasing the number of the minority class or decreasing the number of the majority class in order to obtain approximately the same number of samples for both the classes. This method changes the original distribution of each class.

Under-sampling Under-sampling aims to balance class distribution by eliminating majority class samples. This is done until the majority and minority class samples are balanced out. Since the total number of training data is reduced, it can help improve run time and save storage memory especially when the training data size is huge. It can discard potentially useful information which could be important for building a rule of classifiers. Beside, most data observations are precious, it is quite waste to discard even some of them. Also, the sample chosen by random under-sampling may be biased. It will not be an accurate representative of the population. Thereby, the random under-sampling method leads to inaccurate results with the actual test data set.

Over-sampling Over-sampling increases the number of samples in the minority class by replicating them in order to present a higher representation of the minority class in the sample. Over-sampling usually outperforms under-sampling because this method leads to no information loss. However, the replication of the minority class samples causes overfitting problem.

Synthetic Minority Over-sampling Technique (SMOTE) This technique is followed to avoid overfitting which occurs when exact replicas of minority samples are added to the original dataset. A subset of training set is taken from the minority class as a sample set and then new synthetic similar samples are created from the subset. These synthetic samples are then added to the original training set. The new training set is used as a sample to train the classification models.

Let $\boldsymbol{x}_p$ be a sample in minority class, $\boldsymbol{x}_q$ be another sample in the minority class randomly chosen among the k-nearest neighbors of $\boldsymbol{x}_p$. A new minority sample is generated in the line segment between $\boldsymbol{x}_p$ and $\boldsymbol{x}_q$ as

$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_p + (\boldsymbol{x}_q - \boldsymbol{x}_p) \times \delta \tag{2.1}$$

where δ is randomly chosen from the uniform distribution on $[0, 1]$.

SMOTE solves the problem of overfitting caused by replicas of minority samples and reserves all useful information. However, SMOTE does not take neighboring samples from other classes into consideration. This can generate additional noise samples and can increase the effect of outliers. Also SMOTE is not very effective for high dimensional data and the computation of generating synthetic samples results in increase in run time.

2.1.2 Ensembling Methods

Ensembling method involves constructing several two stage classifiers from the original data and then aggregate their predictions. The main objective of ensemble methods is to improve the performance of single classifiers by modifying existing classification algorithms to make them appropriate for imbalanced data..

2.1.3 Cost-Sensitive Learning

Cost-sensitive learning is a type of learning that takes the misclassification costs into consideration and minimize the total cost. For example, using penalized algorithms to increase the cost of misclassification on the minority class. In this section we introduce several reweighted loss functions to compare with the proposed method.

Softmax In multi-class classification, the output of a neural network has to be normalized into a probability distribution over predicted output classes in order to determine which class it belongs. The softmax function is often used as an activation function of the last layer.

Let $\boldsymbol{z} = (z_1, z_2, \ldots, z_m)$ be the output of a m-classification model, where m is the total number of classes. The probability output after softmax is:

$$y_k = \frac{e^{z_k}}{\sum_{k=1}^{m} e^{z_k}},\ k = 1, \ldots, m, \tag{2.2}$$

where y_k is the probability of the sample belongs to the kth class.

Cross-Entropy (CE) Loss The cross-entropy loss function is a mostly used loss function in classification problems. For the binary classification problems, the target value t_n is zero or one, $t_n \in \{0, 1\}$. Then, the sigmoid activation function is used for the last layer,

$$CE = -[t_n \log(y_n) + (1 - t_n) \log(1 - y_n)], \tag{2.3}$$

where y_n is output of the nth training sample, y_n also represents the probability that this sample is classified to the positive class.

In a multi-classification problem, the softmax activation function is used for the last layer, the cross-entropy loss can be reduced to:

$$CE = -\log(y_n), \tag{2.4}$$

where y_n is the probability that nth sample is classified to the target class. For a batch with an input of N samples, the cross entropy loss is:

$$CE = -\frac{1}{N} \sum_{n=1}^{N} \log(y_n). \tag{2.5}$$

Focal Loss A small number of difficult samples make a large contribution to the total training loss, whereas a large number of simple samples make a small contribution to the total loss. The focal loss enables the network to focus more on minority samples which give a high contribution to the total loss.

The focal loss multiplies a coefficient of $(1 - y_n)^\gamma$ based on the cross-entropy loss function to reduce the relative loss of difficult samples and increase the relative loss of easy samples. The focal loss is shown as follows:

$$FL = -(1 - y_n)^\gamma \log(y_n), \tag{2.6}$$

where $\gamma\,(\gamma > 0)$ is a tuneable focusing parameter. When $\gamma = 0$, the focal loss is formally the same as the cross-entropy loss.

We introduce the α-balanced variable to improve accuracy. The α-balanced focal loss is denoted

as:

$$FL = -\alpha(1 - y_n)^{\gamma} \log(y_n). \tag{2.7}$$

For a batch with input of N samples, the focal loss is:

$$FL = -\frac{1}{N} \sum_{n=1}^{N} \alpha_n (1 - y_n)^{\gamma} \log(y_n). \tag{2.8}$$

CEFL Samples from minority classes tend to have higher loss values than those from majority classes. Cost-sensitive learning methods assign weight for each class according to a given data distrubution. CEFL and CEFL2 are proposed to rebalance the cross-entropy loss and the focal loss.

Mathematically, we weight $(1 - y_n)$ and y_n to the cross-entropy loss function and the focal loss function respectively:

$$CEFL = -(1 - y_n) \log(y_n) - y_n (1 - y_n)^{\gamma} \log(y_n), \tag{2.9}$$

where $\gamma\,(\gamma > 0)$ is a tuneable focusing parameter, as it is in the focal loss. Note that when $\gamma = 0$, CEFL is formally equivalent to thecross-entropy loss.

For a batch with an input of N samples, the CEFL is:

$$CEFL = -\frac{1}{N} \sum_{n=1}^{N} \left[(1 - y_n) \log(y_n) + y_n (1 - y_n)^{\gamma} \log(y_n) \right]. \tag{2.10}$$

When a sample is poorly classified, y_n is relatively smaller and $(1 - y_n)$ is relatively larger, which makes the contribution of the cross-entropy loss in the total loss lager than the focal loss; thus, CEFL loss is closer to the cross-entropy loss, vice versa.

CEFL2 CEFL2 function is as follows:

$$\begin{aligned} CEFL2 = &- \frac{(1 - y_n)^2}{(1 - y_n)^2 + y_n^2} \log(y_n) \\ &- \frac{y_n^2}{(1 - y_n)^2 + y_n^2} (1 - y_n)^{\gamma} \log(y_n). \end{aligned} \tag{2.11}$$

For a batch with an input of N samples, the CEFL2 is:

$$CEFL2 = -\frac{1}{N}\sum_{n=1}^{N}\left[\frac{(1-y_n)^2}{(1-y_n)^2+y_n^2}\log(p) + \frac{y_n^2}{(1-y_n)^2+y_n^2}(1-y_n)^{\gamma}\log(y_n)\right]. \tag{2.12}$$

2.2 Evaluation Metric

The classification result of a binary classification task has four possible situation. False negative (FN) is the number of samples whose classification results are negative but their true labels are positive, false positive (FP) is the number of samples whose classification results are positive but their true labels are negative. True positive (TP) is the number of samples whose classification results are positive and their true labels are positive, true negative (TN) is the number of samples whose classification results are negative and their true labels are negative. As shown in Table 2.1. The evaluation metric we introduced in this section are given by those four values.

Table 2.1: Binary classification

	Classification result	
True label	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

Let $X, \hat{X} \in \{\mathrm{P}, \mathrm{N}\}$ be the true label and the estimated label respectively. Then we define the following evaluation metrics.

- True Positive Rate (TPR) or Sensitivity:

$$\mathrm{Sen} = \mathrm{P}(\hat{X} = \mathrm{P} \,|X = \mathrm{P}) = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{2.13}$$

- True Negative Rate (TNR) or Specificity

$$\mathrm{Spe} = \mathrm{P}(\hat{X} = \mathrm{N} \,|X = \mathrm{N}) = \frac{\mathrm{TN}}{\mathrm{FP}+\mathrm{TN}} \tag{2.14}$$

- False Positive Rate (FPR):

$$\mathrm{FPR} = \mathrm{P}(\hat{X} = \mathrm{P} \,|X = \mathrm{N}) = 1 - \mathrm{Spe} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{2.15}$$

- False Negative Rate (FNR):

$$\mathrm{FNR} = \mathrm{P}(\hat{X} = \mathrm{N} \,|X = \mathrm{P}) = 1 - \mathrm{Sen} = \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}} \tag{2.16}$$

2.2.1 Accuracy, Precision, Recall and F1 Score

- Accuracy (ACC):

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{2.17}$$

  Accuracy is not the best metric when evaluating imbalanced datasets as it can be misleading.

- Precision:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{2.18}$$

  Precision is also called the positive predictive value (PPV). It is a measure of a classifier's exactness. The lower precision indicates the higher number of false positives.

- Recall:

$$\mathrm{Recall} = \mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2.19}$$

  Recall is also called the sensitivity or the true positive rate (TPR). It is a measure of a classifier's completeness. The lower recall indicates the higher number of false negatives.

- F-measure (F_1 score):

$$\mathrm{F}_1 \text{ score} = \frac{2 * \mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} = \frac{2\,\mathrm{TP}}{2\,\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{2.20}$$

F_1 score is the harmonic average of the precision and the recall. It is a measure of a test's accuracy.

2.2.2 ROC and AUC

For binary classification problems, an unknown input $\boldsymbol{x}_n$ is classified to the negative class (N) if a output value y_n is less than a thresholding parameter θ, otherwise is classified to the positive class (P). We define a unit function

$$u\left(y_n - \theta\right) = \begin{cases} 1 & \theta < y_n \\ 0 & \text{otherwise.} \end{cases} \tag{2.21}$$

Then the true positive rate (sensitivity) and the true negative rate (specificity) are given as

$$\text{Sen}(\theta) = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{1}{N_+} \sum_{n=1}^{N} t_n u\left(y_n - \theta\right) \tag{2.22}$$

$$\text{FPR}(\theta) = \frac{\text{FP}}{\text{FP} + \text{TN}} = \frac{1}{N_-} \sum_{n=1}^{N} \left(1 - t_n\right) u\left(y_n - \theta\right) \tag{2.23}$$

where N_+ is the total number of positive samples, N_- is the total number of negative samples, the defination of TP, FP, TN and FN are shown in Table 2.1.

If the prior probabilities of two classes are the same 0.5, it minimizes the classification error when y_n is the posterior probability. By modifying θ, we can control the trade-off between the sensitivity and specificity. If we use small θ, unconfined samples are classified to the positive class, then the sensitivity becomes larger, but the specificity becomes smaller. In contrast, when θ is large, unconfined samples are classified to the negative class, then the sensitivity becomes smaller, but the specificity becomes larger. For example, in the first stage of the diagnosis, we should use smaller θ.

The receiver operating characteristic (ROC) curve is the plot with the false positive rate and the true positive rate as x and y axes as shown in Figure 2.1.

AUC is the area under the ROC curve. AUC takes value from zero to one, and the value is equivalent to the probability that the classifier will rank a randomly chosen positive sample higher

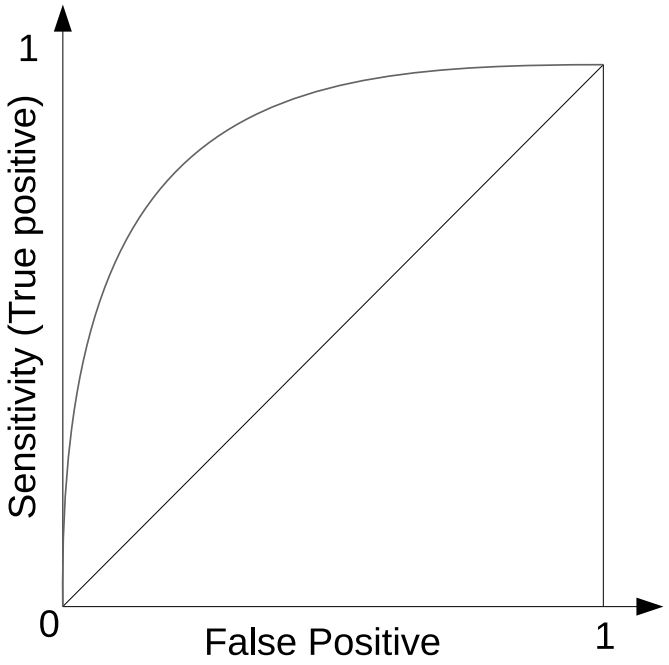

Figure 2.1: The receiver operating characteristic (ROC) curve

than a randomly chosen negative sample [9]. Since the false positive and sensitivity are functions of θ, we denote $\mathrm{FPR}(\theta)$ and $\mathrm{Sen}(\theta)$. Then AUC is given by the Stieltjes integral,

$$\mathrm{AUC} = \int_0^1 \mathrm{Sen}(\theta) d\mathrm{FPR}(\theta) \tag{2.24}$$

Furthermore, if FPR is differentiable,

$$\mathrm{AUC} = \int_0^1 \mathrm{Sen}(\theta) \frac{d\mathrm{FPR}(\theta)}{d\theta} d\theta \tag{2.25}$$

In an alternative interpretation, AUC is the probability,

$$\mathrm{AUC} = P(y_+ \geq y_-), \tag{2.26}$$

where y_+ and y_- are the outputs of the classifier for samples randomly chosen from the positive and negative classes, respectively [8].

2.3 Neural Network

Neural network is one of the most popular machine learning models. It has been widely applied on image recognition task. Most neural networks are designed into layers of nodes and feed-forward structure. In this section, we introduce the conventional deep neural network we used in the experiment for structure data and the convolutional neural network we used for image classification tasks.

2.3.1 Forward Propagation

Let $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^N \subset \mathbb{R}^d$ be a set of d dimensional input data, where N is the number of input data. The bias term is included in the last dimension of d-dimensional space, and we do not consider it explicitly. Let $\{\boldsymbol{t}_n\}_{n=1}^N \subset \mathbb{R}^q$ be the corresponding target data, where the output dimension is q. For unspecified input, we omit the subscript n. For a vector $\boldsymbol{x}$, $[\boldsymbol{x}]_j$ denotes the jth element of $\boldsymbol{x}$. Let $(\boldsymbol{x}_1, \boldsymbol{t}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{t}_N) \in \left(\mathbb{R}^d \times \mathbb{R}^q\right)$ be input data and corresponding target vector. Let I_l, $l = 0, \ldots, L$ be the number of units in the lth layer, where $l_0 = d$ and L is the number of layers. The artificial feed-forward neural network propagates the data from the former layer to the next layer, output of the first layer is $\boldsymbol{z}^1 = \boldsymbol{x}$, then output of the lth layer is $\boldsymbol{z}^{l+1}$

$$\boldsymbol{z}^{(l+1)} = \phi_{(l)}\left({\boldsymbol{W}^{(l)}}^\top \boldsymbol{z}^{(l)}\right),\ l = 1, \ldots, L, \tag{2.27}$$

where $\boldsymbol{W}^{(l)} = \left[\boldsymbol{w}_1^{(l)} | \ldots | \boldsymbol{w}_{I_l}^{(l)}\right] \in \mathbb{R}^{I_{l-1} \times I_l}$ is the weight matrix. $\phi_{(l)}$ is the element-wise activation function of the lth layer. The output of the last layer is $\boldsymbol{y} = \boldsymbol{z}^{(L+1)}$. We implicitly consider the bias term $\boldsymbol{z}^{(l)} \leftarrow \begin{bmatrix} \boldsymbol{z}^{(l)} \\ 1 \end{bmatrix}$ here. Figure 2.2 shows a brief structure of a neural network with one hidden layer.

The activation function we used in the hidden layer is the rectified linear unit (ReLU)

$$\phi_{(l)}(z) = z^+ = \max(0, z). \tag{2.28}$$

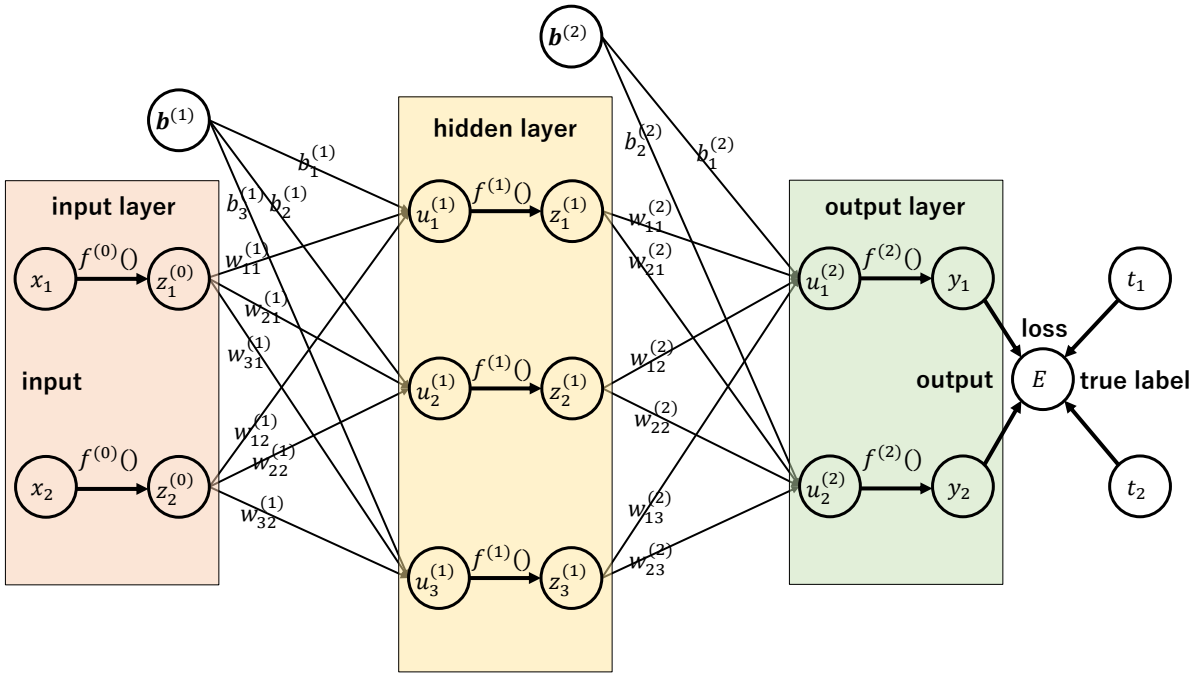

Figure 2.2: Neural Network

2.3.2 Loss Function

The weight matrices $\boldsymbol{W}^{(l)}$ (including bias terms) are tuned to minimize a loss (error) function. Thus, the loss function should be chosen based on the objective of the problem. In the classification problems, logistic loss, cross-entropy loss, or (multi-class) hinge loss are often used.

In the binary classification problems, the logistic loss is given by Eq. (2.3)

$$\begin{aligned} E_n &= t_n \ln y_n + (1 - t_n) \ln (1 - y_n) \\ E &= \sum_{n=1}^{N} E_n, \end{aligned} \tag{2.29}$$

where y_n is output of the nth training sample and t_n is the target value.

In order to perform the back propagation, we determine $\frac{\partial E_n}{\partial y_n}$

$$\frac{\partial E_n}{\partial y_n} = \frac{t_n}{y_n} - \frac{1 - t_n}{1 - y_n} = \frac{t_n - y_n}{y_n (1 - y_n)}. \tag{2.30}$$

In multiclass classification, we use the 1-of-K coding and the soft max loss for K-class multiclass classification. In the 1-of-K coding, if $\boldsymbol{x}_n$ belongs the class k, $\boldsymbol{t}_n$ is a K dimensional vector whose kth element is one, and remaining elements are zero. The activation of the last layer is the softmax loss. Then the probability that the nth sample $\boldsymbol{x}_n$ belongs to the kth class is $[\boldsymbol{y}_n]_k$ is given by Eq. (2.2). Then the loss is given by

$$\begin{aligned} E_n &= \sum_{k=1}^{K} [\boldsymbol{t}_n]_k \ln [\boldsymbol{y}_n]_k \\ E &= \sum_{n=1}^{N} \sum_{k=1}^{K} [\boldsymbol{t}_n]_k \ln [\boldsymbol{y}_n]_k \,. \end{aligned} \tag{2.31}$$

2.3.3 Back-propagation

The back propagation method is used to obtain the weight $\boldsymbol{W}$ that minimizes the loss function E.

$$\boldsymbol{w}_k^{(l)} \leftarrow \boldsymbol{w}_k^{(l)} - \eta \nabla_{\boldsymbol{w}_k^{(l)}} E, \quad k = 1, \ldots, I_l \tag{2.32}$$

where η is the learning rate. For each layer, the gradient is obtained from the chain rule,

$$\nabla_{\boldsymbol{w}_k^{(l)}} E = \sum_{n=1}^{N} \delta_{k,n}^{(l)} \boldsymbol{z}_n^{(l)}, \tag{2.33}$$

$$k = 1, \ldots, I_l,\, l = 1, \ldots, L-1$$

$$\delta_{k,n}^{(L)} = \frac{\partial E_n}{\partial y_n} \phi'_{(L)} \left((\boldsymbol{w}_{k,n}^{(L)})^\top \boldsymbol{z}_{k,n}^{(L)} \right) \tag{2.34}$$

$$\delta_{k,n}^{(l)} = \phi'_{(l)} \left((\boldsymbol{w}_{k,n}^{(l)})^\top \boldsymbol{z}_{k,n}^{(l)} \right) \sum_{h=1}^{I_l} \delta_{h,n}^{(l+1)} \left[\boldsymbol{w}_h^{(l+1)} \right]_k, \; l = 1, \ldots, L-1. \tag{2.35}$$

2.3.4 Convolutional Neural Network

Convolutional neural network (CNN) is a type of deep neural networks, most commonly applied to image recognition. We also apply the proposed method to image classification task on CNN architecture.

Convolutional layer extracts features from an input image matrix, it can preserve the relationship between pixels by learning image features using small squares of input data. Convolution is a

mathematical operation which requires two inputs: a image matrix and a kernel. The size of the square is determined by the convolutional kernel:

$$\boldsymbol{Z}^{l+1} = \sum_{k=1}^{K_l} \sum_{i=1}^{L} \sum_{j=1}^{L} \left(\boldsymbol{Z}_{i,j,k}^{l} \boldsymbol{w}_{k}^{l+1} \right) + \boldsymbol{b}, \tag{2.36}$$

where $\boldsymbol{b}$ is a bias, $\boldsymbol{Z}^l$ and $\boldsymbol{Z}^{l+1}$ are the input and output of the $l+1$th layer which are also called feature map, (i, j) is the pixel index in the feature map, k is used to index the channels of the feature map, $\boldsymbol{Z}_{i,j,k}^{l}$ stands for the input patch centered at location (i, j) from the kth channel in the $l+1$th layer. L is the size of $\boldsymbol{Z}^{l+1}$ and K_l is the number of channels in the lth layer. Then applying activation function

$$\boldsymbol{A}_{i,j,k}^{l} = \phi \left(\boldsymbol{Z}_{i,j,k}^{l} \right). \tag{2.37}$$

Usually, the activation function ϕ is ReLU as Eq. (2.28). Then the output is sent to the pooling layer which can reduce the number of parameters when the images are too large. For example, max pooling takes the largest element from the feature map.

2.3.5 Deep Residual Neural Network

The residual network introduces a shortcut structure and solves the problem of gradient disappearance to some extent with a deepening network layer. As Figure 2.3 shown, $\mathbf{x}$ is the input of the block, $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ is the output of the block, residual $\mathcal{F}(\mathbf{x})$ is what actually trained inside the layers.

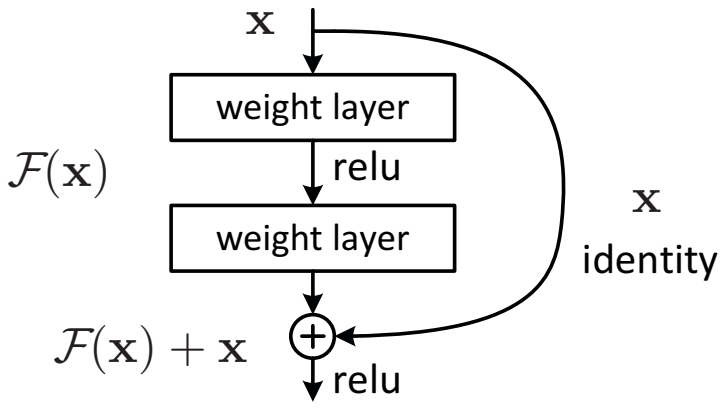

Figure 2.3: Residual learning block [18]

These residual networks are easier to optimize, and can gain better performance from considerably increased depth.

Chapter 3

Proposed Method

3.1 AUC Loss

As we introduced in Chapter 2, AUC is the area under the ROC curve. Since the AUC calculated by Eq.(2.24) is not differentiable, so we estimate the empirical AUC and use it for the loss function of neural network.

3.1.1 Estimation of AUC

AUC is given by the Stieltjes integral,

$$\mathrm{AUC} = \int_0^1 \mathrm{Sen}(\theta) d\mathrm{FPR}(\theta). \tag{3.1}$$

$\mathrm{Sen}(\theta)$ and $\mathrm{FPR}(\theta)$ are given as

$$\mathrm{Sen}(\theta) = \frac{1}{N_+} \sum_{n=1}^{N} t_n u\left(y_n - \theta\right) \tag{3.2}$$

$$\mathrm{FPR}(\theta) = \frac{1}{N_-} \sum_{n=1}^{N} \left(1 - t_n\right) u\left(y_n - \theta\right), \tag{3.3}$$

where N_+ is the total number of positive samples, N_- is the total number of negative samples, and the unit function is

$$u\left(y_n-\theta\right)=\begin{cases}1 & \theta<y_n\\ 0 & \text{otherwise.}\end{cases} \tag{3.4}$$

By substituting Eq. (3.2) and Eq. (3.3) to Eq. (3.1), the estimation of AUC is given by

$$\begin{aligned}\mathrm{AUC}&=\int_0^1 \mathrm{Sen}(\theta)d\mathrm{FPR}(\theta)\\ &=\frac{1}{N_+}\sum_{m=1}^{N} t_m\left(\mathrm{FPR}\left(y_m\right)-1\right).\end{aligned} \tag{3.5}$$

Since the unit function of $\mathrm{FPR}(\theta)$ given by Eq. (3.3) is not differentiable. Thus we approximate $u(\cdot)$ by the sigmoid with a scale parameter α,

$$\begin{aligned}\sigma_\alpha(t)&=\frac{1}{1+\exp(-\alpha t)}\\ \frac{d}{dt}\sigma_\alpha(t)&=\alpha\sigma_\alpha(t)\left(1-\sigma_\alpha(t)\right).\end{aligned} \tag{3.6}$$

As shown in Figure. 3.1, α determines the similarity to $u(\cdot)$, the larger the closer.

In order to maximize the AUC, we simply take $E=-\mathrm{AUC}$ as a loss function, by substituting Eq. (3.6) to Eq. (3.5), we have

$$E=\frac{1}{N_+}\sum_{m=1}^{N} t_m\left(1-\frac{1}{N_-}\sum_{n=1}^{N}\left(1-t_n\right)\sigma_\alpha\left(y_n-y_m\right)\right). \tag{3.7}$$

Since $u(\cdot)$ is replaced by $\sigma_\alpha(t)$, we can obtain $\frac{\partial E}{\partial y_m}$ to calculate the back-propagation update. We now obtain the gradient,

$$\begin{aligned}\frac{\partial E}{\partial y_m}=-\frac{1}{N_+N_-}\Big(&\sum_{i=1,\,i\neq m}^{N} t_i(1-t_m)\frac{\partial\sigma_\alpha(y_m-y_i)}{\partial y_m}\\ &+\sum_{j=1,\,j\neq m}^{N}(1-t_j)t_m\frac{\partial\sigma_\alpha(y_j-y_m)}{\partial y_m}\Big).\end{aligned} \tag{3.8}$$

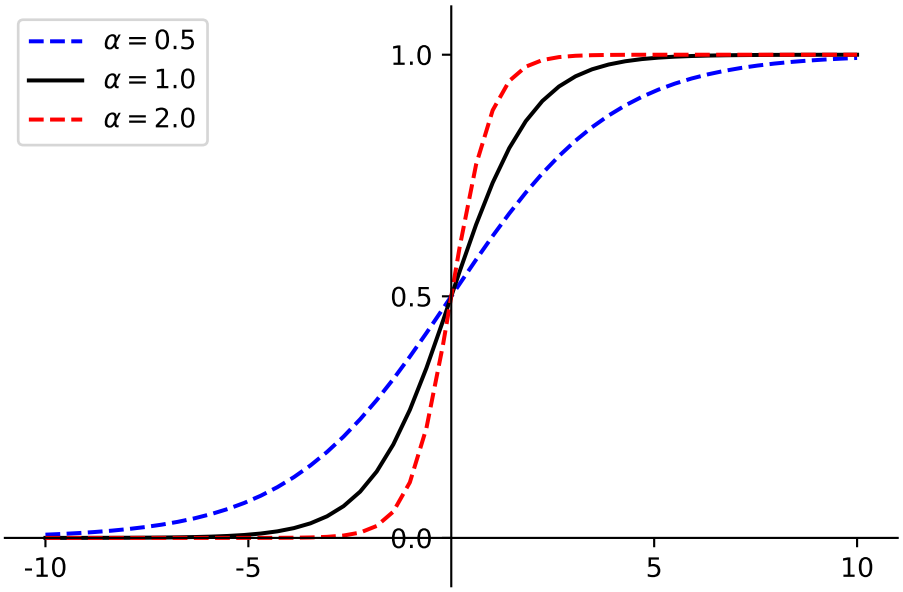

Figure 3.1: Sigmoid

By applying Eq. (3.8) to Eqs. (2.33)-(2.35), we can derive the back-propagation rule to minimize AUC.

It should be noted that AUC loss is not a simple summation for samples, n. Therefore, $\frac{\partial E}{\partial y_n} \neq \sum_{n=1}^{N} \frac{\partial E_n}{\partial y_n}$ because AUC does not make sense for only one training sample.

3.1.2 Sampling Method

The calculation cost of AUC is lager than conventional loss functions, so we are going to apply some sampling method to reduce the run time. t_m and t_n in Eq. (3.7) are given by mini-batch and reservoir sampling separately.

Mini-batch $\mathcal{D} = \left\{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \{-1, +1\} | i \in [N]\right\}$ is an imbalanced dataset. We separate the positive samples and the negative samples into two sets: $\mathcal{D}_+ = \left\{\left(\boldsymbol{x}_i^+, +1\right) | i \in [N_+]\right\}$ and $\mathcal{D}_- = \left\{\left(\boldsymbol{x}_j^-, -1\right) | j \in [N_-]\right\}$, where N_+ and N_- are the number of positive and negative samples respectively.

The imbalanced ratio inside one mini-batch is fixed as $r = \frac{N_-}{N_+}$, the same with the imbalanced ratio of the whole dataset, in order to prevent overfitting. The size of one mini-batch is M, the

number of positive samples and negative samples inside the batch are $M_+ = [M \cdot (1-r)]$ and $M_- = [M \cdot r]$. Since some datasets are heavily imbalanced so the batch size should be larger, for example, if $r = 0.01$ then M must be larger than 100 to make sure that there is at least one negative sample inside the batch considered that both N_+ and N_- of Eq. (3.7) can't be zero.

We randomly choose M_+ and M_- samples from $\mathcal{D}_+$ and $\mathcal{D}_-$ respectively, and put them together as one mini-batch for training. The process of mini-batch is shown in Figure 3.2.

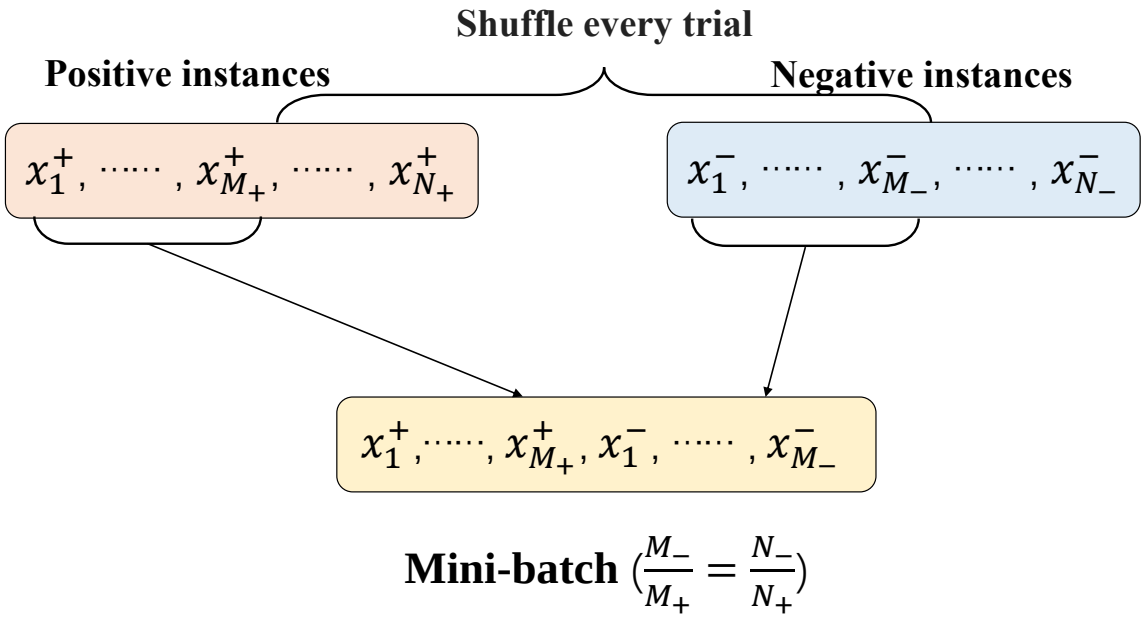

Figure 3.2: Mini-batch

Reservoir sampling Several random sampling methods are applied to AUC maximization problems, the reservoir sampling and online AUC maximization framework [11, 12]. The main purpose of this method is caching a small number of received training examples which maintain an accurate a sketch of the whole data under the constraint of fixed buffer size.

We introduce two buffers, B_+ and B_- of size N_+ and N_-, for storing the received positive and negative samples. $(\boldsymbol{x}_t, y_t)$ is a sample received at trial t, we update the two buffers by comparing $\boldsymbol{x}_t$ to samples in B_+^t if $y_t = 1$ and to samples in B_-^t if $y_t = -1$. Then use the two buffers to train the model.

The buffers are updated after each trial of training. Given a received training example $(\boldsymbol{x}_t, y_t)$,

add it to the buffer $B^t_{y_t}$ if $\left|B^t_{y_t}\right| < N_{y_t}$. Otherwise, with probability $\frac{N_{y_t}}{N^{t+1}_{y_t}}$, we update the buffer $B^t_{y_t}$ by randomly replacing one sample in $B^t_{y_t}$ with $\boldsymbol{x}_t$. The samples in the buffers simulate a uniform sampling from the original data. Algorithm 1 shows the process of updating the buffers with reservoir sampling [11].

Algorithm 1 UpdateBuffer with Reservoir Sampling [11]

Input

- B^t: the current buffer
- $\boldsymbol{x}_t$: a training sample
- N: the buffer size
- N_{t+1}: the number of samples received till trial t

Output: updated buffer B^{t+1}

```
if |B^t| < N then
  B^{t+1} = B^t ∪ {x_t}
else
  Sample Z from a Bernoulli distribution with Pr(Z = 1) = N/N_{t+1}
  if Z=1 then
    Randomly delete an sample from B^t
    B^{t+1} = B^t ∪ {x_t}
  end if
end if
return B^{t+1}
```

3.2 Multi-class AUC

Receiver operating characteristic (ROC) curves have become a common analysis tool for diagnostic of a binary classifier system. But ROC curve is limited to prediction involving only two possible classes. However, many diagnostic problems exist have multiple possible classes, which are not binary. Some extensions of the ROC curve to multiclass have been explored. For example, the full extension extends ROC curve into high-dimensional hypersurfaces that cannot be visualized but it presents some problems. Therefore, among several different approximations to the full extension, the approach we chosen is a two-class approximations to multiclass problems [16].

In this approach, each observation is classified as either belonging or not belonging to class i,

$i = 1, 2, \ldots, n$ and a ROC curve is produced for each class. An AUC value is calculated for each class. The final AUC is the weighted average of the AUC values of each class.

$$\mathrm{AUC}_{\mathrm{Multi-class}} = \sum_{i=1}^{n} \mathrm{AUC}\,(i)\, p\,(i) \tag{3.9}$$

where $\mathrm{AUC}\,(i)$ is the AUC of class i, and $p\,(i)$ is the prior probability of class i. The advantage of this approach is that the calculation complexity of the multiclass problem increase linearly with number of classes n. The results are easy to visualize, because there is one ROC curve for each class. As a evaluation metric, this multi-class AUC value is affected by the distribution of each class. But for AUC maximization, we define the loss function as $E = -\mathrm{AUC}_{\mathrm{Multi-class}}$. As an optimization object, this affect is acceptable.

Chapter 4

Experiments and Results

We conducted experiments to demonstrate the proposed method and compared with the conventional method on binary data classification and multi-class image classification tasks.

4.1 Implementation

The neural network we used for binary classification task has 3 hidden layers with 128 units in each hidden layer. The activate functions of the hidden layers and the output layer were the rectified linear unit (ReLU) and sigmoid function respectively. The model was trained by mini-batch with 128 batch-size, 128 buffer-size, 4000 iteration, $\alpha = 10$ and learning rate $\eta = 0.001$. We used five-fold cross-validation with random data splitting. We used one of the most widely applied network architecture, ResNet-50 as the neural network for image classification task. All experiments were implemented by Python 3.7 and all models are based on PyTorch. We used a stochastic gradient descent (SGD) optimizer and ran on PC having Core i7-9900K with 32 GB memory and rtx 2060.

4.2 Imbalanced Datasets

We used synthesis and open datasets of binary classification problems. We generated 6 imbalanced synthesis datasets data 1-6. Each dataset has 5000 samples and 100 features. They are generated by Gaussian distribution with 6 different negative rate 0.1%, 0.5%, 1%, 5%, 10%, 35%. The real

datasets are downloaded from LIBSVM website[1] [19] and UCI (University of California, Irvine) machine learning repository[2]. We used balanced datasets breast cancer and sonar, we also reduced the number of negative sample in these datasets into 30 to make imbalanced datasets imbalanced breast cancer and imbalanced sonar. The information of the datasets we used in experiment are shown in Table 4.1.

Table 4.1: Datasets statistics

Name	Features	Samples	Negative rate	Positive rate
data1	100	5000	0.001	0.999
data2	100	5000	0.005	0.995
data3	100	5000	0.01	0.99
data4	100	5000	0.05	0.95
data5	100	5000	0.10	0.90
data6	100	5000	0.35	0.65
satimage	36	6430	0.09	0.91
breast cancer	30	569	0.38	0.62
sonar	60	208	0.50	0.50
cryotherapy	6	90	0.47	0.53
skin-nonskin	3	245057	0.20	0.80
immunotherapy	7	90	0.22	0.78

The datasets we used in image classification task were handwritten digit (USPS and MNIST), the most widely used CIFAR-10 and the street view house numbers (SVHN). We conducted a certain amount of random sampling on the original CIFAR-10 and MNIST training set for each class to obtain a class-imbalanced training set. To measure the imbalance of the new imbalanced CIFAR-10 and MNIST training set, we introduce the imbalance ratio ρ of the numbers of samples between the class with the most samples and the class with the fewest samples [20].

$$\rho = \frac{\max_i \{|C_i|\}}{\min_i \{|C_i|\}}, \tag{4.1}$$

[1]https:// www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/
[2]http://archive.ics.uci.edu/ml/

where C_i is a set of samples in class i, $\max_i \{|C_i|\}$ is the number of samples in the class with the most samples, and $\min_i \{|C_i|\}$ is the number of samples in the class with the fewest samples. To evaluate the performance on different imbalance degree, a set of ρ values ($\rho \in \{1,10,20,50,100\}$) were evaluated.

4.3 Binary Classification

The information of the datasets we used in experiment binary classification are shown in Table 4.1. Table 4.2 shows the comparison of AUC, classification accuracy and run time with full-batch training. The proposed method performed higher AUC than the conventional method on all imbalanced datasets, since the datasets are imbalanced, the comparision of accuracy is meaningless. Especially on imbalanced datasets like data3, data4 and imbalanced sonar, the proposed method achieved better performance than the logistic loss with higher AUC values. This confirmed the advantage that optimizing AUC as loss function fits better on imbalanced distributed datasets in our theory. Besides the improvement of AUC and accuracy on imbalanced datasets, the computational complexity of the proposed method is higher than the conventional method. This is because that the calculation of AUC requires the nested summation in Eq. (3.7) for all samples in the batch set.

Table 4.2: Comparison of AUC, accuracy and run time of full-batch

Data	AUC loss			Logistic loss		
	AUC	Accuracy [%]	Time [s]	AUC	Accuracy [%]	Time [s]
data3	0.7551±0.0621	97.00±0.75	439.4	0.7241±0.0740	98.48±0.01	16.9
data4	0.8867±0.0232	95.94±0.58	449.7	0.8673±0.0293	96.44±0.46	17.3
data5	0.9335±0.0150	95.12±0.67	436.9	0.9282±0.0163	95.50±0.60	16.9
data6	0.9766±0.0039	93.32±0.62	446.2	0.9723±0.0029	92.72±0.27	17.3
satimage	0.9534±0.0024	89.66±0.75	565.2	0.9494±0.0015	93.47±0.01	488.7
breast cancer	0.9954±0.0047	97.01±1.00	11.1	0.9955±0.0045	97.19±0.73	3.4
imbalanced breast cancer	0.9986±0.0025	98.36±1.63	7.7	0.9978±0.0042	98.52±1.35	2.4
sonar	0.7437±0.1584	65.44±11.75	5.1	0.7522±0.1539	67.34±0.09	2.2
imbalanced sonar	0.9691±0.0386	88.95±10.69	3.7	0.9602±0.0267	90.52±8.76	1.9

To reduce the run time, we applied mini-batch and reservoir sampling as mentioned in Section 3.1.2. Table 4.3 shows the comparison of AUC, classification accuracy, F-Score and run time with

mini-batch training. The proposed AUC loss achieved better AUC value on most datasets, especially on heavily imbalanced dataset data1-3. As we supposed, mini-batch learning can avoid local minimum which means better performance and heavily reduce the run time. The run time of the proposed AUC loss is still longer than the conventional logistic loss, but comparing with full-batch situation in Table 4.2, the difference is acceptable.

Table 4.3: Comparison of AUC, accuracy and run time of mini-batch

Data	AUC loss				Logistic loss			
	AUC	Accuracy [%]	Time [s]	F-Score	AUC	Accuracy [%]	Time [s]	F-Score
data1	0.8676±0.1117	99.04±0.67	31.7	0.9951	0.6777±0.0310	99.08±0.01	21.1	0.9954
data2	0.9178±0.0995	99.14±0.68	22.2	0.9887	0.8518±0.0388	98.78±0.01	15.9	0.9940
data3	0.9256±0.0870	98.84±0.95	22.3	0.9941	0.8353±0.0506	98.30±0.01	15.8	0.9914
data4	0.9726±0.0222	98.26±0.75	17.2	0.9883	0.9725±0.0157	96.76±0.45	12.5	0.9876
data5	0.9784±0.0115	97.24±0.97	22.8	0.9847	0.9879±0.0030	97.18±0.69	20.9	0.9845
data6	0.9861±0.0035	95.28±0.92	23.5	0.9640	0.9943±0.0025	96.28±0.45	16.7	0.9718
satimage	0.9452±0.0150	86.87±0.56	22.1	0.5689	0.9380±0.0109	90.50±0.30	15.5	0.0424
breast cancer	0.9962±0.0073	97.54±0.73	21.7	0.9804	0.9911±0.0075	97.37±1.24	15.5	0.9790
imbalanced breast cancer	0.9902±0.0136	98.37±1.77	36.5	0.9916	0.9916±0.0153	97.28±0.01	22.9	0.9862
sonar	0.7864±0.1522	69.30±11.91	21.3	0.6891	0.7967±0.1339	69.29±13.27	15.1	0.7142
imbalanced sonar	0.9847±0.0139	96.28±2.08	23.3	0.9794	0.9847±0.0139	95.32±3.37	13.4	0.9746

Table 4.4 shows the comparison of AUC, classification accuracy, F-Score and run time with reservoir sampling. Reservoir sampling heavily reduced run time as we supposed, but the performance on imbalanced situation is not as good as we supposed. The reason is that the buffers are update one at a time, which makes the buffers are almost the same between each training iteration. This online update framework doesn't work well on imbalanced situation because the update probability of negative buffer and positive buffer are the same.

Table 4.4: Comparison of AUC, accuracy and run time of reservoir sampling

Data	AUC loss				Logistic loss			
	AUC	Accuracy [%]	Time [s]	F-Score	AUC	Accuracy [%]	Time [s]	F-Score
data1	0.5400±0.0673	80.00±1.38	8.3	0.8886	0.4976±0.0987	86.56±2.13	6.9	0.9277
data2	0.6432±0.0687	90.36±2.06	18.2	0.9491	0.6754±0.0681	94.76±0.94	10.8	0.9730
data3	0.6731±0.0665	73.88±1.21	11.3	0.8482	0.7140±0.0309	80.82±2.17	7.5	0.8928
data4	0.8231±0.0448	84.24±2.13	46.2	0.9107	0.8243±0.0225	86.12±1.08	32.5	0.9224
data5	0.9009±0.0141	88.66±1.38	52.2	0.9345	0.8933±0.0207	89.90±1.37	29.7	0.9421
data6	0.8894±0.0081	83.86±0.61	53.5	0.8769	0.8786±0.0197	82.32±1.74	24.7	0.8675
satimage	0.9158±0.0116	85.61±1.92	38.8	0.5247	0.9172±0.0136	87.03±2.01	25.7	0.5387
breast cancer	0.9945±0.0039	96.83±1.35	11.5	0.9747	0.9942±0.0050	97.54±0.74	7.9	0.9804
imbalanced breast cancer	0.9916±0.0139	98.45±0.58	13.0	0.9916	0.9865±0.0118	98.19±0.71	8.2	0.9903
sonar	0.7363±0.1698	65.38±14.07	5.3	0.6517	0.7427±0.1366	65.42±12.35	4.0	0.6576
imbalanced sonar	0.9345±0.0412	88.95±7.72	4.2	0.9229	0.9550±0.0401	89.72±8.30	3.2	0.9282

The computational complexity of the proposed method is higher than the conventional method, because that the calculation of AUC requires the nested summation in Eq. (3.7). The samples given this nested summation which are t_n and t_m have no need to be the same. So, we determine t_m by mini-batch and t_n from buffers of reservoir sampling. Table 4.5 shows the comparison of AUC, classification accuracy, F-Score and run time of this combination method of mini-batch and reservoir sampling. We also compared the proposed AUC loss with combination of SMOTE and binary cross entropy (BCE) using mini-batch learning.

Table 4.5: Training with mini-batch and reservoir sampling

Data	AUC loss		Logistic loss		SMOTE and BCE	
	AUC	F-Score	AUC	F-Score	AUC	F-Score
data1	0.8624±0.0995	0.9953±0.0029	0.6785±0.0588	0.9952±0.0002	0.7517±0.0236	0.7088±0.0001
data2	0.9069±0.0903	0.9885±0.0017	0.8549±0.0471	0.9940±0.0000	0.7664±0.0155	0.9433±0.0042
data3	0.9239±0.0893	0.9946±0.0046	0.8374±0.0505	0.9914±0.0000	0.9457±0.0327	0.8861±0.0350
data4	0.9726±0.0190	0.9881±0.0032	0.9725±0.0114	0.9865±0.0024	0.9664±0.0057	0.9919±0.0025
data5	0.9737±0.0116	0.9842±0.0046	0.9847±0.0045	0.9825±0.0039	0.9746±0.0105	0.9876±0.0023
data6	0.9873±0.0030	0.9653±0.0073	0.9940±0.0025	0.9718±0.0045	0.9959±0.0033	0.9671±0.0067
satimage	0.9473±0.0135	0.9046±0.0078	0.9379±0.0133	0.9489±0.0002	0.9315±0.0116	0.9356±0.0094
breast cancer	0.9962±0.0048	0.9818±0.0080	0.9914±0.0074	0.9776±0.0087	0.9946±0.0041	0.9778±0.0072
sonar	0.7979±0.1436	0.6828±0.1206	0.7887±0.1374	0.7083±0.1104	0.7478±0.1110	0.6569±0.1346
cryotherapy	0.9851±0.0161	0.9447±0.0395	0.9846±0.0175	0.9692±0.0285	0.9527±0.0393	0.8687±0.0738
skin-nonskin	0.9544±0.0346	0.8732±0.0737	0.9497±0.0431	0.8409±0.0692	0.9847±0.0354	0.8857±0.0681
immunotherapy	0.7459±0.1188	0.8532±0.0284	0.7761±0.1035	0.8677±0.0354	0.6629±0.1024	0.8448±0.0501

Figures 4.1 and 4.2 are the comparison of AUC value and F-score from Table 4.5. The performance of AUC loss is better than both logistic loss and combination of SMOTE and binary cross entropy on most imbalanced dataset. Even on those balance dataset like sonar and cryotherapy, the performance is still better than conventional methods.

As we mentioned, calculation time is always a big problem for the proposed method, Figure 4.3 and Table 4.6 show the comparison of run time from Table 4.5. Since we combined the mini-batch and reservoir sampling to reduce the run time and avoid local minimum, the difference of run time

Table 4.6: Run time [s] with mini-batch and reservoir sampling

Data	data1	data2	data3	data4	data5	data6	satimage	breast cancer	sonar	cryotherapy	skin-nonskin	immunotherapy
AUC loss	49.7	31.5	31.6	32.0	31.2	31.6	24.4	26.7	16.5	32.7	33.1	33.8
Logistic loss	34.2	23.3	23.3	24.2	23.5	23.8	23.3	19.9	15.3	21.3	25.8	29.5
SMOTE and BCE	88.4	100.8	81.1	97.0	88.3	87.4	89.8	70.9	70.3	96.6	279.2	98.6

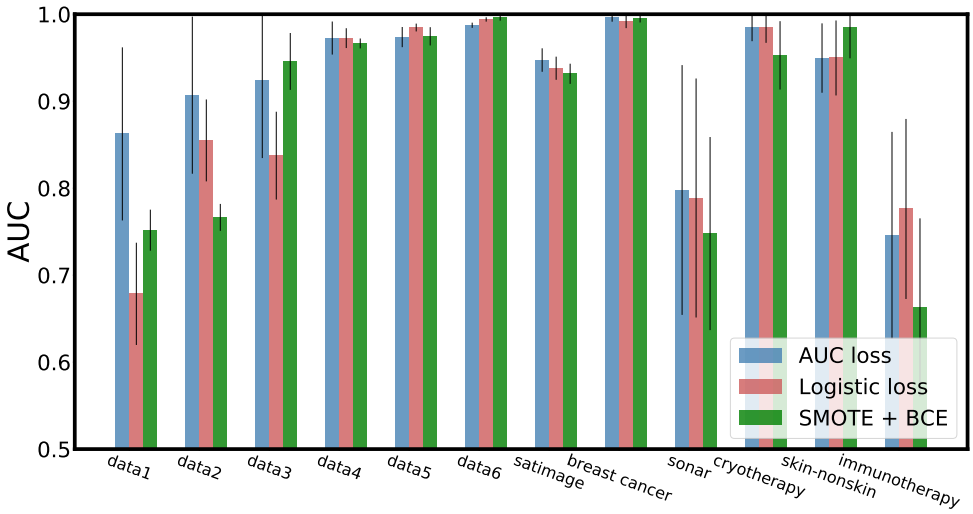

Figure 4.1: AUC

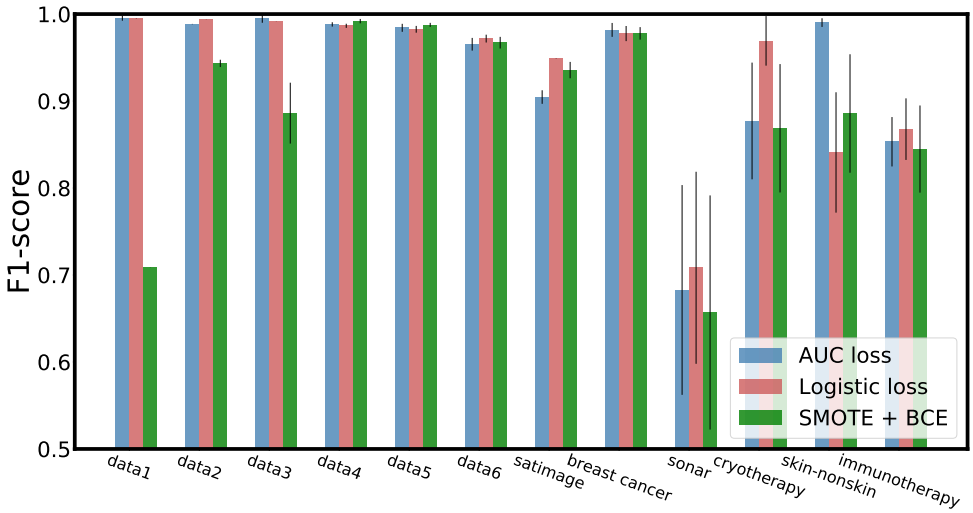

Figure 4.2: F_1-score

between AUC loss and logistic loss is close. But the combination of SMOTE and binary cross entropy takes much longer run time than we supposed. The reason is the algorithm of SMOTE takes a lot of time to generate synthetic samples, especially in heavily imbalanced situation, the amount of synthetic samples needed to generate could be large.

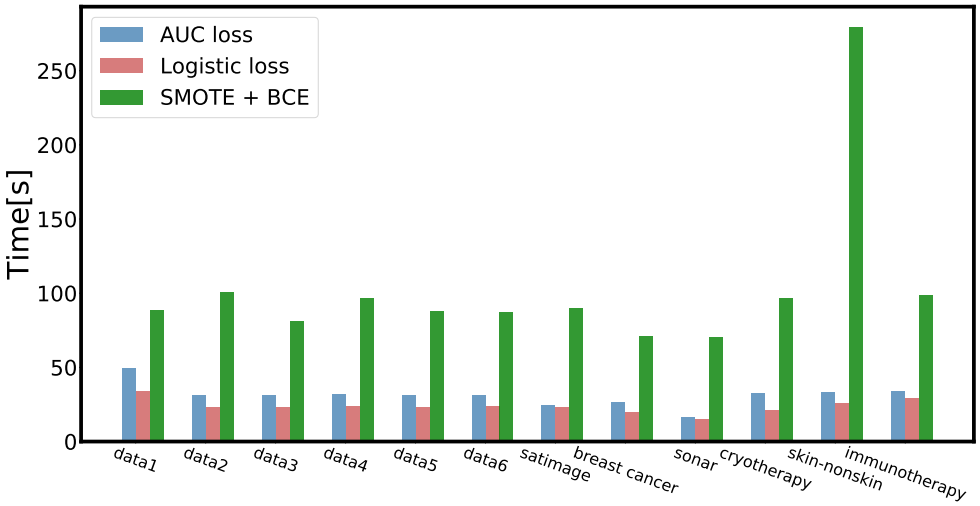

Figure 4.3: Run time

4.4 Image Classification

We compared the performance of the proposed AUC loss with the cross entropy and the focal loss [3] on image classification problem with ResNet-50 model.

Table 4.7 shows the comparison of AUC, classification accuracy, F-Score of the multi-class AUC loss, the cross entropy loss and the focal loss. Since the datasets are mostly balanced, the advantage of AUC loss is not clear, but it still achieved better AUC value due to the optimization of AUC on most datasets.

Table 4.7: Comparison of AUC, accuracies and F_1-score

Data	Multi-class AUC loss			Cross entropy loss			Focal loss		
	AUC	Accuracy	F_1-score	AUC	Accuracy	F_1-score	AUC	Accuracy	F_1-score
CIFAR10	0.9028	0.5526	0.5440	0.8881	0.4935	0.4890	0.8450	0.4076	0.3869
MNIST	0.9998	0.9856	0.9854	0.9998	0.9883	0.9882	0.9998	0.9866	0.9865
QMNIST	0.9995	0.9829	0.9827	0.9970	0.9200	0.9184	0.9994	0.9702	0.9698
FashionMNIST	0.9917	0.8859	0.8825	0.9925	0.8937	0.8933	0.9924	0.8922	0.8924
USPS	0.9928	0.9187	0.9100	0.9933	0.9332	0.9279	0.9277	0.9952	0.9216
EMNIST	0.9811	0.7920	0.6986	0.9959	0.7943	0.7194	0.9953	0.7759	0.6973
SVHN	0.9929	0.9114	0.9032	0.9910	0.8916	0.8826	0.9901	0.8887	0.8775

We manually conducted imbalanced CIFAR-10 and imbalanced MNIST with different imbalance ratio ρ ($\rho \in \{1,10,20,50,100\}$). The results are shown in Table 4.8 and Table 4.9. As the imbalance ratio goes larger, the advantages of AUC loss and the focal loss become obvious. The proposed AUC loss achieved better performance on imbalanced CIFAR-10 and almost comparable performance against the focal loss. The focal loss is also a method for imbalanced situation, but there are two more hyper-parameters α and γ to tune which means it is complicated in practice application.

Table 4.8: Comparison of AUC, accuracies and F_1-score on imbalanced CIFAR10

Imbalanced Ratio	10			20			50			100		
Metrics	AUC	Accuracy	F_1-score	AUC	Accuracy	F_1-score	AUC	Accuracy	F_1-score	AUC	Accuracy	F_1-score
Cross-Entropy Loss	0.8536	0.4537	0.4489	0.8360	0.4140	0.3922	0.780	0.3132	0.3056	0.7439	0.2873	0.2677
Focal Loss(α=0.25,γ=2)	0.8663	0.4415	0.4215	0.8148	0.3289	0.3142	0.7369	0.2997	0.2671	0.7579	0.2919	0.2922
Multi-Class AUC Loss	0.8639	0.4698	0.4625	0.8380	0.4116	0.4054	0.8093	0.3563	0.3418	0.7598	0.3104	0.3003

Table 4.9: Comparison of AUC, accuracies and F_1-score on imbalanced MNIST

Imbalanced Ratio	10			20			50			100		
Metrics	AUC	Accuracy	F_1-score	AUC	Accuracy	F_1-score	AUC	Accuracy	F_1-score	AUC	Accuracy	F_1-score
Cross-Entropy Loss	0.9988	0.9616	0.9613	0.9963	0.9085	0.9068	0.9789	0.8234	0.8230	0.9513	0.6107	0.5868
Focal Loss(α=0.25,γ=2)	0.9983	0.9410	0.9400	0.9946	0.9119	0.9113	0.9853	0.8136	0.8122	0.9628	0.6799	0.6730
Multi-Class AUC Loss	0.9989	0.9537	0.9531	0.9959	0.9207	0.9203	0.9837	0.8439	0.8419	0.9601	0.6465	0.6352

Table 4.10: Average run time [s]

Data	Imbalanced CIFAR10	Imbalanced MNIST
Cross-Entropy Loss	147.1	56.6
Focal Loss(α=0.25,γ=2)	144.1	53.1
Multi-Class AUC Loss	145.1	53.7

Chapter 5

Conclusion

5.1 Discussion

In binary classification task, we conducted the experiment to compare the performance of the proposed AUC loss and the conventional logistic loss. The advantage of AUC loss is the optimization of AUC, since accuracy is not a reliable measure due to the imbalance, higher AUC value means better performance. But the calculation of AUC loss costs a lot of time, that makes computation complexity become the biggest disadvantage of it. To solve the problem, we considered two approach, mini-batch learning and reservoir sampling. The experimental result shows that the reservoir sampling heavily reduces run time but at a cost of performance. The mini-batch training does not reduce run time as much as the reservoir sampling but the performance becomes better. Since AUC loss is a nested summation, we combine the mini-batch learning and the reservoir sampling together. The samples for training are given by mini-batch and the calculation of AUC loss are given by both. This method solves the problem of run time and increases the performance. In image classification task, we use the most simple method weighted average to extend the binary form AUC loss into multi-class form to deal with multi-class image classification problems. We choose this method because the other extension of multi-class AUC such as pairwise form comes with much higher computation complexity, which makes the calculation time problem more serious. But still, our proposed multi-class AUC loss has comparable performance against the focal loss.

5.2 Conclusion

In this paper, we have presented a novel loss function using the AUC criterion for a binary classification neural network and extended it into multi-class form for ResNet framework. First, we formulated AUC by the Stieltjes integral and the unit function, and approximated it by using the sigmoid function with a scale parameter. Then we derived the differentiation of AUC and update rule to the back-propagation algorithm. We conducted numerical experiments to compare our proposed with the conventional method. In order to address the disadvantage of computation complexity, we combined the mini-batch learning and the reservoir sampling method to evaluate the AUC loss to avoid local minimum and reduce the calculation time. Also we extended our method by weighted average to multi-class classification problems and applied ResNet framework to multi-class image classification task. As a result, optimizing AUC as the loss function performs better especially on imbalanced classification tasks.

5.3 Future Work

Future tasks include; i) investigating another approximation method of AUC instead of using sigmoid, in order to reduce the number of hyper-parameters; ii) using better extension of multi-class AUC without increasing run time.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[2] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[3] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[5] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[6] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme. Cost-sensitive learning methods for imbalanced data. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010.

[7] L. Wang, C. Wang, Z. Sun, S. Cheng, and L. Guo. Class balanced loss for image classification. *IEEE Access*, 8:81142–81153, 2020.

[8] J.A. Hanley and B. Mcneil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 05 1982.

[9] T. Fawcett. Introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 06 2006.

[10] H. Ghanbari and K. Scheinberg. Directly and efficiently optimizing prediction error and AUC of linear classifiers. *arXiv:1802.02535*, 02 2018.

[11] P. Zhao, S. Hoi, R. Jin, and T. Yang. Online AUC maximization. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 233–240, 01 2011.

[12] W. Gao, R. Jin, S. Zhu, and Z. Zhou. One-pass auc optimization. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–906–III–914. JMLR.org, 2013.

[13] K. Ren, H. Yang, Y. Zhao, M. Xue, H. Miao, S. Huang, and J. Liu. A robust auc maximization framework with simultaneous outlier detection and feature selection for positive-unlabeled classification. *IEEE Transactions on Neural Networks and Learning Systems*, PP:3027–3083, 03 2018.

[14] Y. Ying, L. Wen, and S. Lyu. Stochastic online AUC maximization. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 451–459, 2016.

[15] M. Liu, X. Zhang, Z. Chen, X. Wang, and T. Yang. Fast stochastic AUC maximization with $o(1/n)$-convergence rate. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3189–3197, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[16] T. Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine Learning*, 31:1–38, 01 2004.

[17] K. Kawaguchi, J. Huang, and L. P. Kaelbling. Effect of depth and width on local minima in deep learning. *Neural Computation*, 31(7):1462–1498, 2019.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[19] C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[20] M. Buda, A. Maki, and M. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106, 10 2017.